Oficina UNECT Jr. e DACOMP

PYTHON 3

O que vamos aprender hoje ?

- O que é o Python
- Vantagens e desvantagens
- Python2 x Python3
- Empresas que usam
- Mercado de Trabalho
- Identação/Sintáxe
- “Hello World”
- Declaração de variáveis
- Input de dados
- Tipos de dados básicos
- Casting
- Listas, dicionários e tuplas
- Estruturas de decisão
- Estruturas de repetição
- Funções
- Funções Build-in
- Módulos Python
- Criação de um módulo
- Criação de uma Classe
- Pacotes Python
- Aonde Python me salvou?
- Sua vez!

1. Informações sobre Python

O que é? Como vive? É de comer?

Características do Python

- Linguagem de programação de alto nível
- Interpretada
- Orientada a objetos
- Funcional
- Tipagem dinâmica e forte (?)

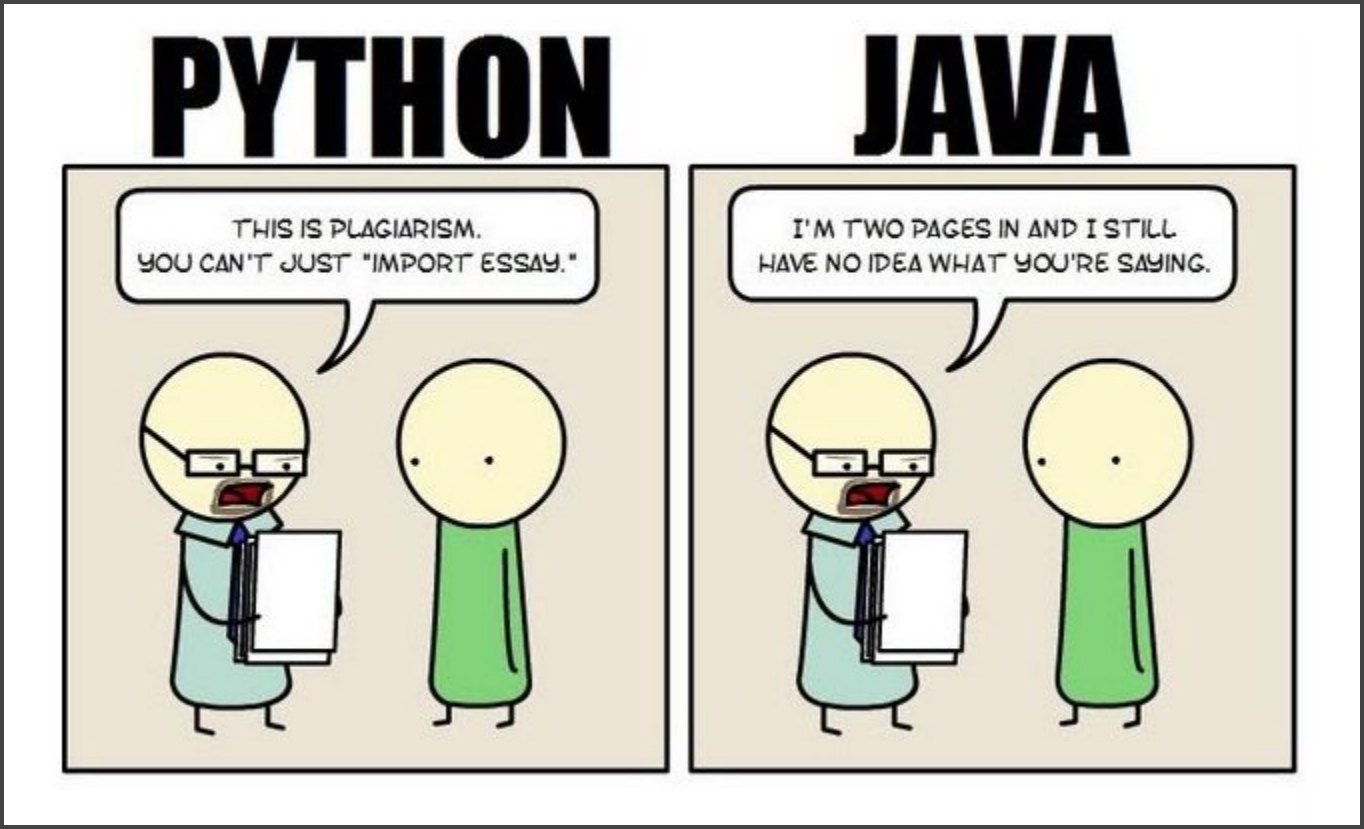
Características do Python

```
>>> x = 'uma string'
>>> y = 2
>>> print(x+y)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to
str implicitly
```

Vantagens do Python

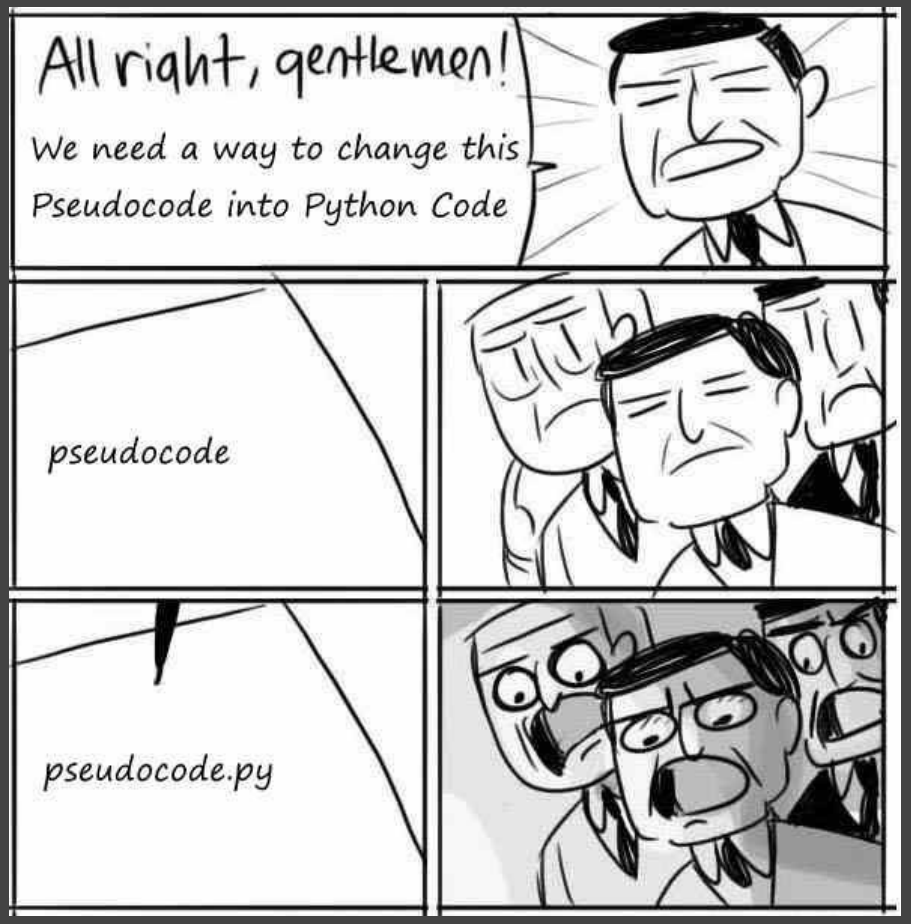
- **Python é fácil de usar/aprender**
- **Multiplataforma**
- **Comunidade ativa**
- **Grande quantidade de bibliotecas de aprendizado computacional, big data, análise estatística, computação numérica...**

Vantagens do Python



Vantagens do Python

- **Legibilidade do código**



Vantagens do Python

- **Legibilidade do código**

Pseudocode

```
if the score is 90 or above
      grade is an "A"
else
      if the score is 80 or above
            grade is a "B"
      else
            if the score is 70 or above
                  grade is a "C"
            else
                  grade is an "F"
```

Actual Python code

```
if score >= 90:
    grade = "A"
else:
    if score >= 80:
        grade = "B"
    else:
        if score >= 70:
            grade = "C"
        else:
            grade = "F"
```

Desvantagens do Python

- **Nenhuma**

Desvantagens do Python

- **Problemas de desempenho em algumas tarefas de alto poder de processamento.**
- **As versões 2 e 3 do Python são incompatíveis entre si, em alguns casos.**
- **Existem outras linguagens melhores para fazer aplicações Desktop, Web e Mobile.**

Python2 e Python3

- Criação
  - Python2 - 2000
  - Python3 - 2008
- Python2 - Última atualização em 17/12/2016
- “Python 2.x é legado, Python 3.x é o presente e o futuro da linguagem” - DOCUMENTAÇÃO, Python.

Python2 e Python3

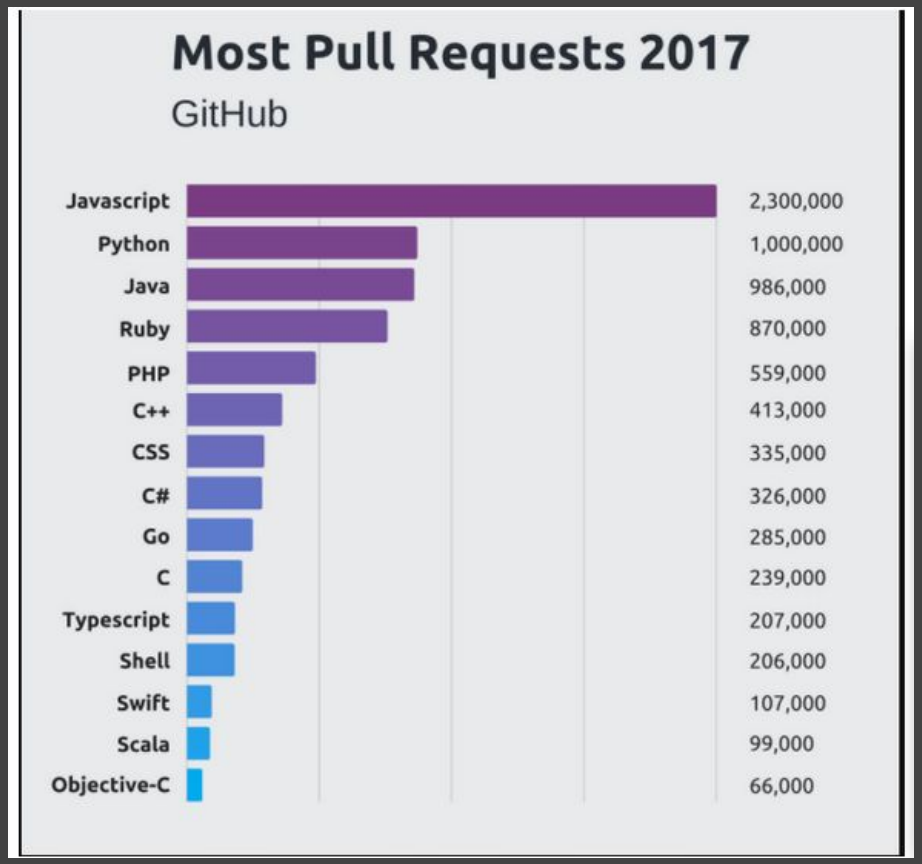
- Python2 vai acabar em 2020, na PyCon
- https://pythonclock.org/

Quem utiliza Python ?



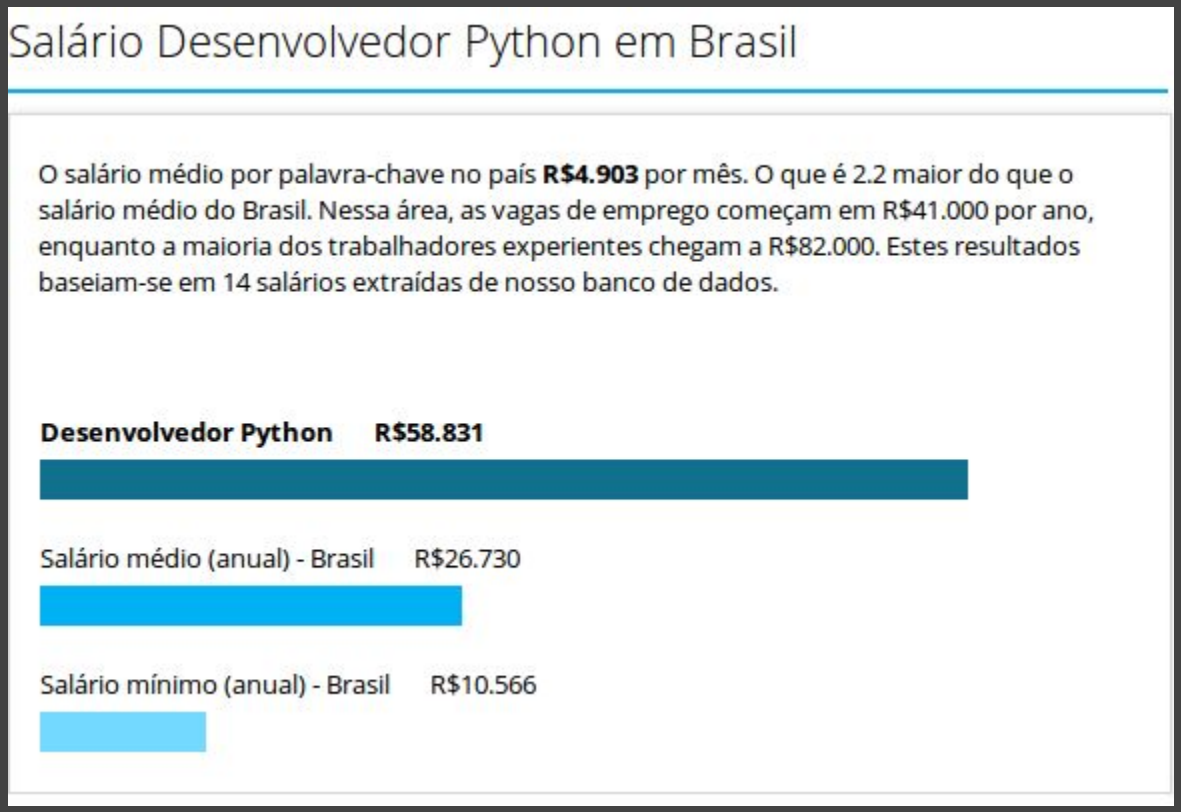



Mercado de Trabalho



https://stackify.com/popular-programming-languages-2018/

Mercado de Trabalho



https://neuvoo.com.br/salario/sal%C3%A1rio-Desenvolvedor-Python

Mercado de Trabalho

- **Empregos Python Brasil**
  - http://www.pyjobs.com.br
  - https://neuvoo.com.br/empregos/?k=python
  - https://www.catho.com.br/vagas/programador-python/
  - https://br.linkedin.com/jobs/python-vagas

2.
Básico do Python

Bora para o código ?

Sintaxe

- **Indentação não é importante, é essencial.**
- **Nunca mais você vai esquecer o ponto e vírgula! (porque não tem)**
- **A vida é muito curta para ficar abrindo e fechando chaves em tudo ...**

Sintaxe - Exemplo

```
if(users):
    all_users = []
    cursor = user.find({})
    for x in cursor:
        x.pop("_id")  # Remove objects id
        all_users.append(x)
    return all_users
```

Hello World - Exemplo

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello world");
    }
}
```

Hello World - Exemplo

```
>>> print("Hello world")
hello world
```

Hello World - Exercício

- Crie um arquivo chamada “hello.py”
- Insira na primeira linha
  - print(“hello world”)
- Execute o programa criado
  - python3 hello.py

Tipos básicos de dados

- **Os quatro tipos básicos mais utilizados são**
  - **Int - Tipo Inteiro - Ex: 3**
  - **Float - Tipo “decimal” - Ex: 3.60**
  - **Str - Tipo string - Ex: “Klebinho”**
  - **Boolean - Tipo Booleano - Ex: True**

Tipos básicos de dados

- **Existem quatro tipos básicos**
  - **Int - Tipo Inteiro - Ex: 3**

```
var tipo_inteiro = 3
```

Tipos básicos de dados

- **Existem quatro tipos básicos**
  - **Float - Tipo “decimal” - Ex: 3.60**

```
var tipo_float = 3.6
```

Tipos básicos de dados

- **Existem quatro tipos básicos**
  - **Str - Tipo string - Ex: “Klebinho”**

```
var tipo_inteiro = "Klebinho"
```

Tipos básicos de dados

- **Existem quatro tipos básicos**
  - **Boolean - Tipo “booleano” - Ex: True**

```
var tipo_bool = True
```

Tipos de dados - Int e Float

- **Operações**

Operação	Operador
adição	+
subtração	-
multiplicação	*
divisão	/

Operação	Operador
exponenciação	**
parte inteira	//
módulo	%

Tipos de dados - Int e Float

```
#soma
>>> 5+5
10
```

```
#multiplicação
>>> 2*2
4
```

```
#subtração
>>> 10-2
8
```

```
#divisão
>>> 81/9
9.0
```

Tipos de dados - Int e Float

```
#exponenciação
>>> 2**5
32

#extração da parte
inteira da divisão
>>> 10//9
1
```

Tipos de dados - Int e Float

Exercícios

- Crie duas variáveis, uma do tipo ‘int’ e a outra ‘float’
- Mostre na tela quatro operações entre elas da sua escolha

Tipos de dados - Int e Float

Exercícios

Formatando para o print

```
## Método "OK"
>>> print('var 1:',2)
var 1: 2
## Método "Gambiarra"
>>> print('var 1:'+ str(2))
var 1:2
```

Formatando para o print

```
## Método "Certo"
>>> print('var 1: {} - var 2: {} - var 3:
{}'.format(2,3.213213213,'umastring'))

var 1: 2 - var 2: 3.213213213 - var 3: umastring

>>> print('var 1: {:4d} - var 2: {:.2f} - var 3:
{:.4}'.format(2,3.213213213,'umastring'))

var 1:    2 - var 2: 3.21 - var 3: umas
```

Tipos de dados - String

- **Strings no python são consideradas uma SEQUÊNCIA**
- **Leia a frase acima, e não esqueça dela.**
- **Além de Strings, Listas, Tuplas e Buffers também são considerados sequências**

Operações em sequências

```
>>> coca = 'Coca'
>>> coca = coca + ' cola'
>>> coca
'Coca cola'
>>> len(coca)
9
```

Operações em sequências

```
>>> coca[0]
'C'
>>> coca[3]
'a'
>>> coca[4]
' '
>>> coca[-1]
'a'
```

Operações em sequências

```
>>> coca.index('C')
0
>>> coca.index('c')
2
>>> coca.count('c')
2
```

Operações em sequências

```
p = "python"
p[:1] # 'p'
p[:2] # 'py'
p[:3] # 'pyt'
p[:4] # 'pyth'
p[:5] # 'pytho'
p[:6] # 'python'
```

```
p = "python"
p[:]  # 'python'
p[1:] # 'ython'
p[2:] # 'thon'
p[3:] # 'hon'
p[4:] # 'on'
p[5:] # 'n'
p[6:] # ''
```

Operações em sequências

```
>>> p = 'python'
>>> p[1:5]
'ytho'
>>> p[1:4]
'yth'
>>> p[2:4]
'th'
```

Operações com String

```
>>> coca.split(" ")
['Coca', 'cola']
>>> coca.split("c")
['Co', 'a ', 'ola']
>>>coca.replace("Coca","Pepsi"
)
'Pepsi cola'
```

Tipos de dados - String
Exercícios

- Crie uma variável chamada “nome_completo” e informe um nome e sobrenome (Ex: Vinicius Silva)
- Com o nome completo dessa pessoa, mostre na tela:
  - A quarta e a ultima letra.
  - As três primeiras letras.
  - Uma lista com nome e sobrenome separados (a partir da string)
  - Contar a quantidade de letras (sem contar o “espaço”)
  - As duas ultimas letras. (DESAFIO)

Input de dados

- **Para obter algum valor de variável, basta chamar a função “input()”.**
- **Passe como parâmetro a mensagem que você quer que apareça na tela.**

Input de dados

```
>>> variavel = input("Digite o que está
sentindo\n")
Digite o que está sentindo
Sono
>>> variavel
'Sono'
>>> type(variavel)
<class 'str'>
```

Input de dados - Exercícios

- Peça uma frase qualquer para o usuário
- Mostre na tela:
  - A frase inserida
  - A quantidade de letras ‘a’

Input de dados - Problema

- **Sempre que utilizamos o input(), ele nos retorna uma variável do tipo “String”.**
- **Nem sempre nós queremos uma string, às vezes precisamos de outros tipos ..**

Casting

- **Para realizar uma mudança de tipo de variável, basta fazer:**
  - **tipo_requerido(variável)**

Casting

```
>>> valor = 2
>>> type(valor)
<class 'int'>
>>> valor
2
>>> str(valor)
'2'
```

```
>>> str_valor = str(valor)
>>> str_valor
'2'
>>> int(str_valor)
2
```

Casting - Exercícios

- Peça dois números para o usuário
- Mostre na tela a soma dos dois números

Estruturas de decisão

- **“If else”**

```
>>> cond = True
>>> if(cond):
...     print("era amor")
... else:
...     print("era cilada")
...
era amor
```

Estruturas de decisão

- **“If else”**

```
>>> cond = False
>>> if(cond):
...     print("era amor")
... else:
...     print("era cilada")
...
era cilada
```

Estruturas de decisão - Condições

Symbol	Meaning
==	Equals
> <	Greater than, less than
>= <=	Greater and less than, or equal to
!=	Not equal
in	Is a value in a list
is	Are the same object*

Estruturas de decisão - Operadores Booleanos

Boolean operator	Boolean operation	Result
and	`False and False`	`False`
	`True and True`	`True`
	`True and False`	`False`
or	`False or False`	`False`
	`True or True`	`True`
	`True or False`	`True`
not	`not True`	`False`
	`not False`	`True`

Estruturas de decisão

- **“If elif else”**

```
nota = 5.6
>>> if(nota <= 5.7):
...     print("Nem queria passar mesmo")
... elif(nota >= 5.8 and nota < 6.0):
...     print("Bora chorar nota para o professor")
... else:
...     print("Achei fácil")
Nem queria passar mesmo
```

Estruturas de decisão

- **“If elif else”**

```
nota = 5.8
>>> if(nota <= 5.7):
...     print("Nem queria passar mesmo")
... elif(nota >= 5.8 and nota < 6.0):
...     print("Bora chorar nota para o professor")
... else:
...     print("Achei fácil")
Bora chorar nota para o professor"
```

Estruturas de decisão

- **“If elif else”**

```
nota = 7
>>> if(nota <= 5.7):
...     print("Nem queria passar mesmo")
... elif(nota >= 5.8 and nota < 6.0):
...     print("Bora chorar nota para o professor")
... else:
...     print("Achei fácil")
Achei fácil
```

Estruturas de decisão - Exercícios

- **Faça um programa que tenha como entrada 2 notas, P1 e P2, faça a média e retorne ao usuário uma mensagem seguindo o critérios de aprovação da UTFPR, as mensagens deverão ser: ‘Aprovado’, ‘Quebra de Pré Requisito’ e ‘Reprovado’.**

Estruturas de repetição

- **Existem duas estruturas de repetição na linguagem Python**
  - **While()**
  - **For()**

While

- Estrutura do while em python, é semelhante às outras linguagens
- O bloco de código irá se repetir até que a condição seja “False”.

While

```
>>> x = 0
>>> while(x<5):
...     print(x)
...     x+=1 # x=x+1
0 1 2 3 4
```

While - Exercício

- Faça um programa que fique pedindo uma palavra para o usuário, até que a palavra inserida tenha mais que duas letras ‘a’ (verificar apenas letras minúsculas).

For

- A estrutura For em Python percorre SEQUÊNCIAS. (Strings, Listas, Tuplas, Buffers..)
- A estrutura é um pouco diferente das outras linguagens, mas depois que se acostuma com a sintaxe, fica muito mais fácil!

For - Sintaxe

```
for elemento in sequencia:
    print(elemento)
```

For - Range

- **Caso eu necessite realizar um For em uma sequência de números específica, posso utilizar a função range()**
- **Possui no máximo três argumentos:**
  - `range(start,stop,step)`

For - Range

```
>>> range(5)
range(0, 5)
>>> list(range(5))
[0, 1, 2, 3, 4]
```

For - Range

```
>>> list(range(-3))
[]
>>> list(range(-3,0))
[-3, -2, -1]
```

For - Range

```
>>> list(range(-2,10,2))
[-2, 0, 2, 4, 6, 8]
>>> list(range(1,10,2))
[1, 3, 5, 7, 9]
```

For - Range

```
>>> list(range(-2,10,2))
[-2, 0, 2, 4, 6, 8]
>>> list(range(1,10,2))
[1, 3, 5, 7, 9]
```

For - Range

```
>>> for x in range(3):
...     print(x)
...
0
1
2
```

For - Range

```
>>> for x in range(3):
...     print(x)
0
1
2
```

For - Range - Exercícios

- Faça um exercício que imprima N números ímpares começando do 1.
- “N” será informada pelo usuário

Listas, Tuplas e Dicionários

- **Utilizadas para armazenar um conjunto de dados**
- **Cada uma tem vantagens e desvantagens, cabe ao programador usá-las de forma correta**

Listas, Tuplas e Dicionários

- **Lista**
  - **“Uma lista é um conjunto ordenado de valores, onde cada valor é identificado por um índice”. Representa uma sequência.**
  - **Utiliza-se [] para sua criação**
  - **Ex: lista = [‘abc’, 1, 2, [‘a’]]**

Listas, Tuplas e Dicionários

- **Tupla**
  - **Uma tupla tem as mesmas características de uma lista, porém é imutável**
  - **Mais rápida que a lista**
  - **Utiliza-se () para sua criação**
  - **Ex: tupla = (2,3,4)**

Listas, Tuplas e Dicionários

- **Dicionário**
  - **Ele é um tipo de mapeamento nativo do Python (Chave: valor)**
  - **Índice é imutável, valor não.**
  - **Utiliza-se {} para sua criação**
  - **Ex: dict_pt_en = { “carro”: “car”, “azul”:”blue”}**

Listas e tuplas - acesso de dados

```
>>> ex_lista = [2,3,'string',[2]]
>>> ex_lista
[2, 3, 'string', [2]]
>>> ex_lista[0]
2
>>> ex_lista[1]
3
```

Listas e tuplas - acesso de dados

```
>>> ex_lista[-1]
[2]
>>> ex_lista[2]
'string'
>>> ex_lista[2][4]
'n'
```

Listas e tuplas - acesso de dados

- **Como uma listas e tuplas são sequências, quaisquer operações em sequências podem ser realizadas. Slides 37-40.**

Listas - manipular dados

```
>>> ex_lista.append(5)
>>> ex_lista
[2, 3, 'string', [2], 5]
>>> ex_lista.remove(2)
>>> ex_lista
[3, 'string', [2], 5]
```

Listas - manipular dados

```
>>> ex_lista.pop(1)
>>> ex_lista
[3, [2], 5]

>>> ex_lista.pop()
>>> ex_lista
[3, [2]]
```

For - Relembrando a sintaxe

```
for elemento in sequencia:
    print(elemento)
```

For - String

```
>>> string = 'teste'
>>> for x in string:
...     print(x)
t e s t e
```

For - Lista

```
>>> lista = [2,3,'string']
>>> for x in lista:
...     print(x)
2
3
string
```

For - Tupla

```
>>> tupla = [2,3,'string']
>>> for x in tupla:
...     print(x)
2
3
string
```

Listas e tuplas - Exercícios

- Altere o exercicio do slide 57:
  - O usuário deve entrar 4 notas que devem ser armazenadas em um lista
  - A partir da lista criada, faça a média simples das notas.

Eu consigo percorrer uma Lista/Tupla e obter o index dela ?

Listas - Valor e Index

- O método correto é utilizar o Enumerate
  - http://book.pythontips.com/en/latest/enumerate.html
- Existe uma forma de fazer com que já foi aprendido até agora, porém não é mais recomendável.

Listas - Valor e Index

- **Isso é claramente uma adaptação técnica...**
-
```
>>> lista = ['a','b','c']
>>> for x in range(len(lista)):
...     print(x,lista[x])
0 a
1 b
2 c
```

Dicionários - acesso de dados

```
>>> dict = {}
>>> dict['blue'] = 'azul'
>>> dict['red'] = 'vermelho'
>>> dict
{'blue': 'azul', 'red': 'vermelho'}
```

Dicionários - sobrescrever dados

```
>>> dict['blue']
'azul'
>>> dict['blue'] = 'Azul'
>>> dict['blue']
'Azul'
```

Dicionários - deletar dados

```
>>> del dict['Azul']
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'Azul'
>>> del dict['blue']
>>> dict
{'red': 'vermelho'}
```

Dicionários com listas

```
>>> telefones = {}
>>> telefones['Klebinho'] = ['40028922','2345678']
>>> telefones
{'Klebinho': ['40028922', '2345678']}
>>> telefones.get('Klebinho')
['40028922', '2345678']
>>> telefones['Klebinho']
['40028922', '2345678']
```

Dicionários com listas

```
>>> len(telefones)
1
>>> len(telefones['Klebinho'])
2
```

Dicionários - Propriedades

```
>>> telefones['Juninho'] = ['99119911']
>>> telefones
{'Juninho': ['99119911'], 'Klebinho': ['40028922',
'2345678']}
>>> telefones.keys()
dict_keys(['Juninho', 'Klebinho'])
```

Dicionários - Propriedades

```
>>> telefones.values()
dict_values([['99119911'], ['40028922',
'2345678']])
>>> telefones.items()
dict_items([('Juninho', ['99119911']),
('Klebinho', ['40028922', '2345678'])])
```

For - Dicionários

```
>>> dict = {1:"um",2:"dois"}
>>> for x in dict:
...     print(x)
1
2
```

For - Dicionários

```
>>> dict = {1:"um",2:"dois"}
>>> for x in dict:
...     print(dict[x])
um
dois
```

For - Dicionários

```
>>> dict = {1:"um",2:"dois"}
>>> for key,value in dict.items():
...     print(key,value)
1 um
2 dois
```

For - Dicionários - Exercícios

- Crie o seguinte dicionário
  - notas = {'aluno1': [6, 7], 'aluno2': [9, 8], 'aluno3': [3, 4]}
- Faça com que o usuário insira mais uma nota em cada aluno. Verifique qual aluno ficou com a maior média (Faça a verificação se a nota é válida).

Verificação em String, Listas, Tuplas e Dicionários

```
>>> string = 'uma string'
>>> lista = ['uma', 'lista']
>>> tupla = ('uma', 'tupla')
>>> dict = {"um":"dicionario"}
```

Verificação em String, Listas, Tuplas e Dicionários

```
>>> "uma" in string
True
>>> "um" in string
True
>>> "uma" in lista
True
>>> "um" in lista
False
```

Verificação em String, Listas, Tuplas e Dicionários

```
>>> "uma" in tupla
True
>>> "um" in tupla
False
>>> "uma" in dict
False
>>> "um" in dict
True
```

Verificação - Exercícios

- Peça ao usuário que insira três nomes em uma lista
- Após a lista ser criada, mostre na tela se existe algum nome igual a “Klebinho”.

Funções

- **“Um dos grandes benefícios é não precisar copiar o código todas as vezes que precisar executar aquela operação, além de deixar a leitura do código mais intuitiva.”**

Definindo uma função sem retorno

```
>>> def funcao(parametro):
...     print(parametro)
>>> funcao('oi')
oi
>>> funcao('olá')
olá
```

Definindo uma função com retorno

```
>>> def funcao(parametro):
...         return parametro*2
>>> x = funcao(2)
>>> x
4
```

Funções - Exercícios

- Crie uma função que recebe um valor em dinheiro. Retorne quantas vezes pode-se comer uma refeição no RU na UTFPR-CP com o valor inserido.

Funções Build-in

- O interpretador do python já tem várias funções prontas para utilizar.
- Talvez uma função que você precise, já está pronta.

Funções Build-in

2. Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.

		Built-in Functions		
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

Módulos

- Programas que são feitos para serem reaproveitados
- Pode conter variáveis, funções, classes, mas todas com uma funcionalidade em comum
- Ex: Um programa que clareia uma imagem.

Pacotes Python

- Um pacote é uma coleção de módulos.
- Pacotes são estruturados dentro de pastas.
- Ex: Um pacote para edição de imagens
  - Dentro dele existem vários módulos cada um respectivo por uma funcionalidade

Gerenciador de Pacotes

- Podemos utilizar pacotes criados por outras pessoas
- O gerenciador de pacotes mais utilizado no python é o PIP
  - https://pypi.org/project/pip/

Criando/usando um módulo

- **Dentro de um arquivo “calculadora.py” coloquei o seguinte código**

```
def soma(x,y):
    return x+y
```

Criando/usando um módulo

- **Para utilizar a função soma, do módulo calculadora.py, basta importar**
-
```
>>> import calculadora
>>> calculadora.soma(2,3)
5
```

Módulo - Exercício

- Crie o módulo apresentado no exemplo acima
- Adicione a função de subtração
- Crie outro arquivo para chamar o módulo “calculadora.py” e realize as operações

Criando uma classe

- **Em um arquivo “calculadora_classe.py”**

```
class Calculadora(object):

    def soma(self, x, y):
        return x + y
```

Criando uma classe

- **Em Python, você não pode criar um método dentro sem o “self”**
  - **Motivo: É uma convenção para deixar explícito a instância do objeto dentro da classe.**

Utilizando uma classe

```
>>> import calculadora_classe
>>> x = calculadora_classe.Calculadora()
>>> x
<calculadora_classe.Calculadora object at
0x7f4c61fc1780>
>>> x.soma(2,3)
5
```

Classe - Exercício

- Crie a classe do exemplo acima.
- Crie a função de multiplicação para a classe Calculadora.
- Crie outro arquivo para utilizar os métodos da classe.

3.

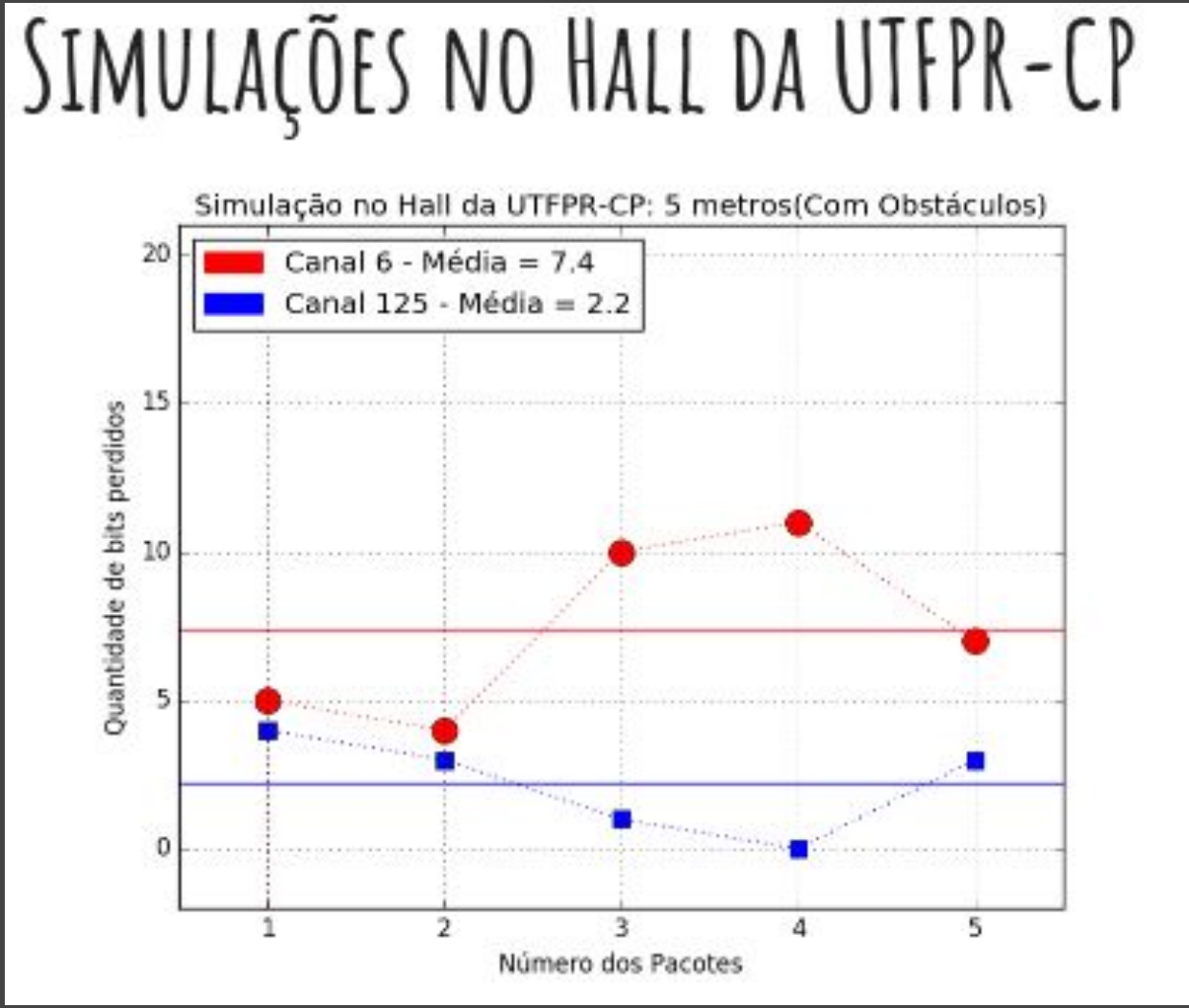
Aonde Python me salvou?

Algumas matérias da
faculdade que utilizei python

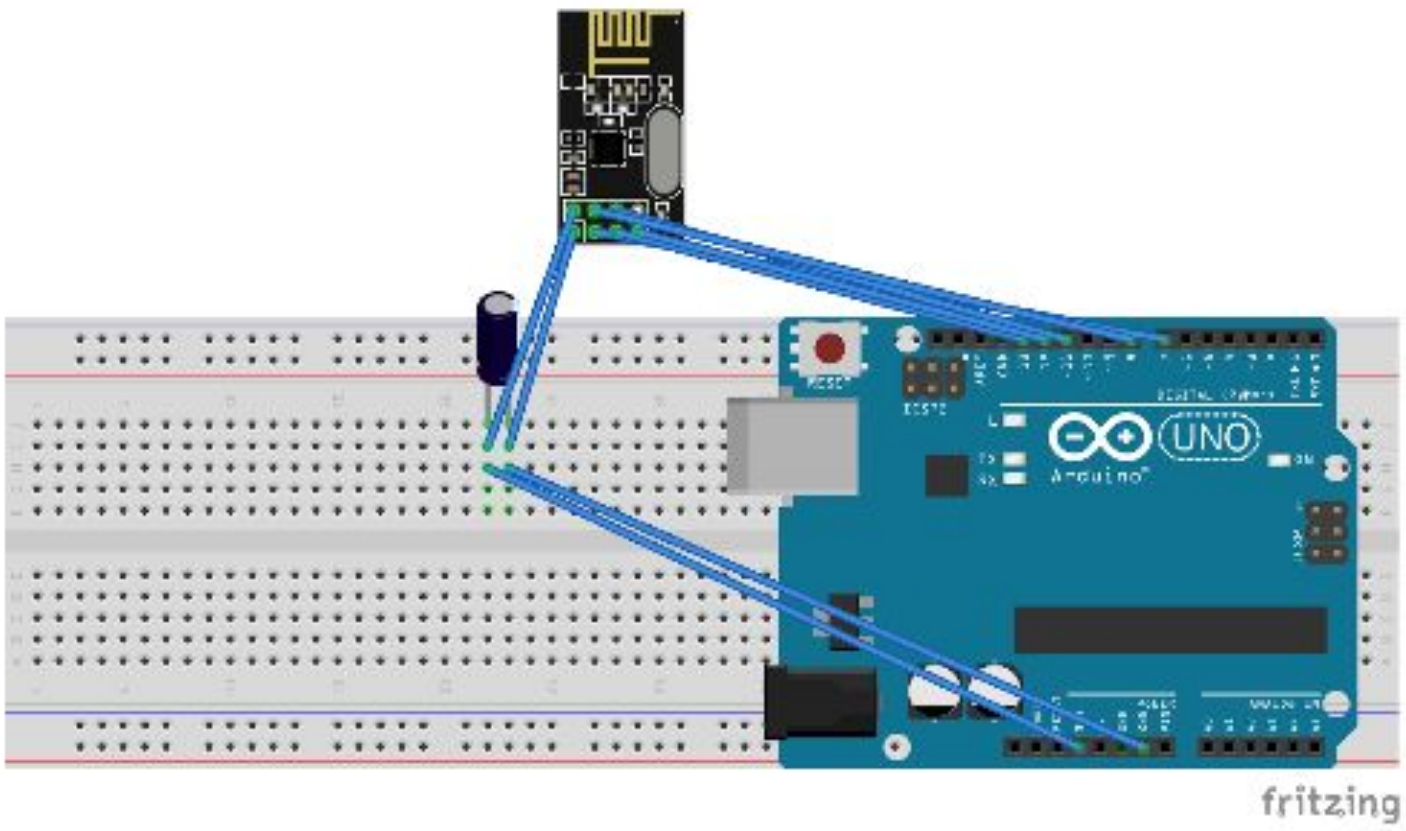
Banco de Dados 2

- **Projeto final de banco de dados**
  - **C++ = 400 linhas ~**
  - **Python = 180 linhas**
- **nested-loop-blocado-bufferizado**

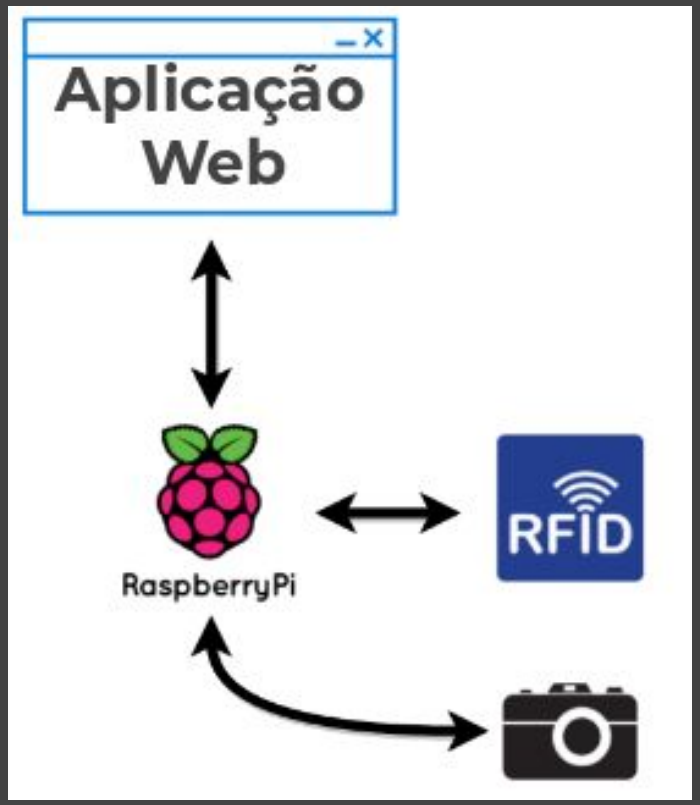
Transmissão de dados



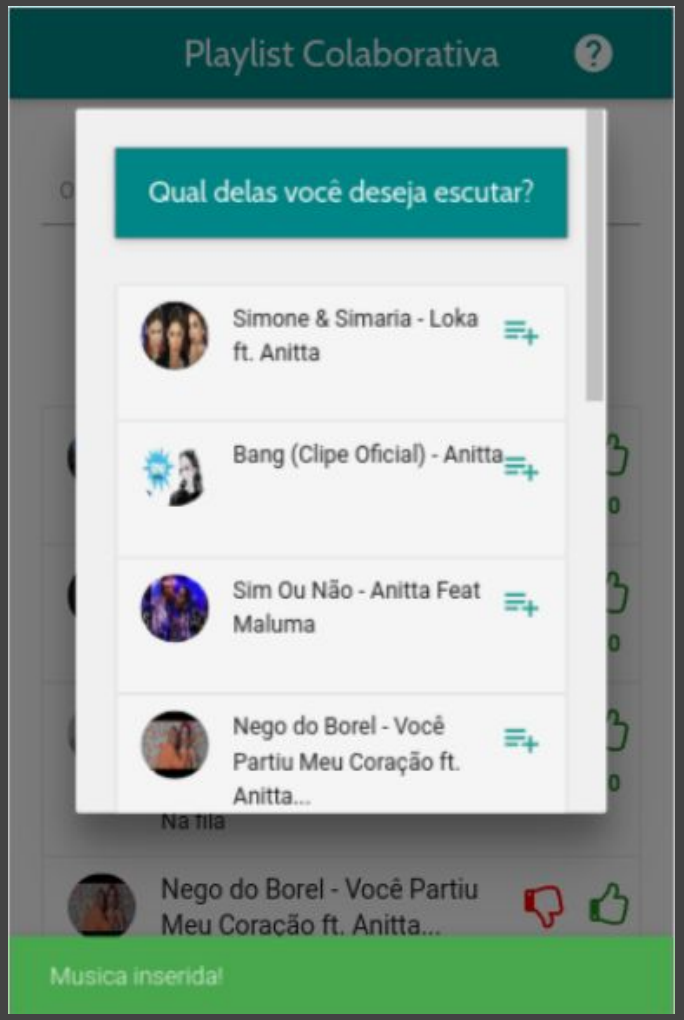
Transmissão de dados



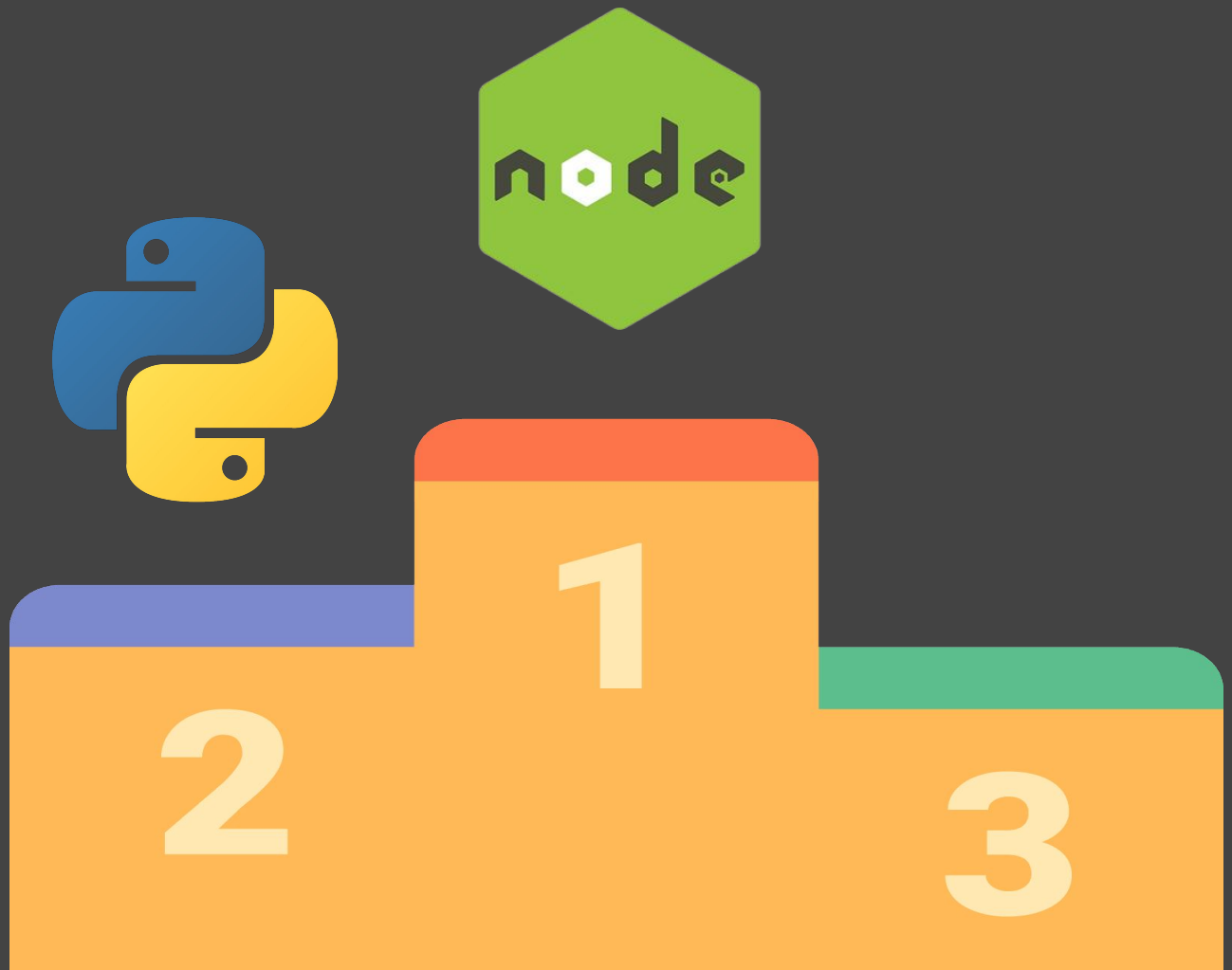
Oficina de integração 1



Oficina de integração 2

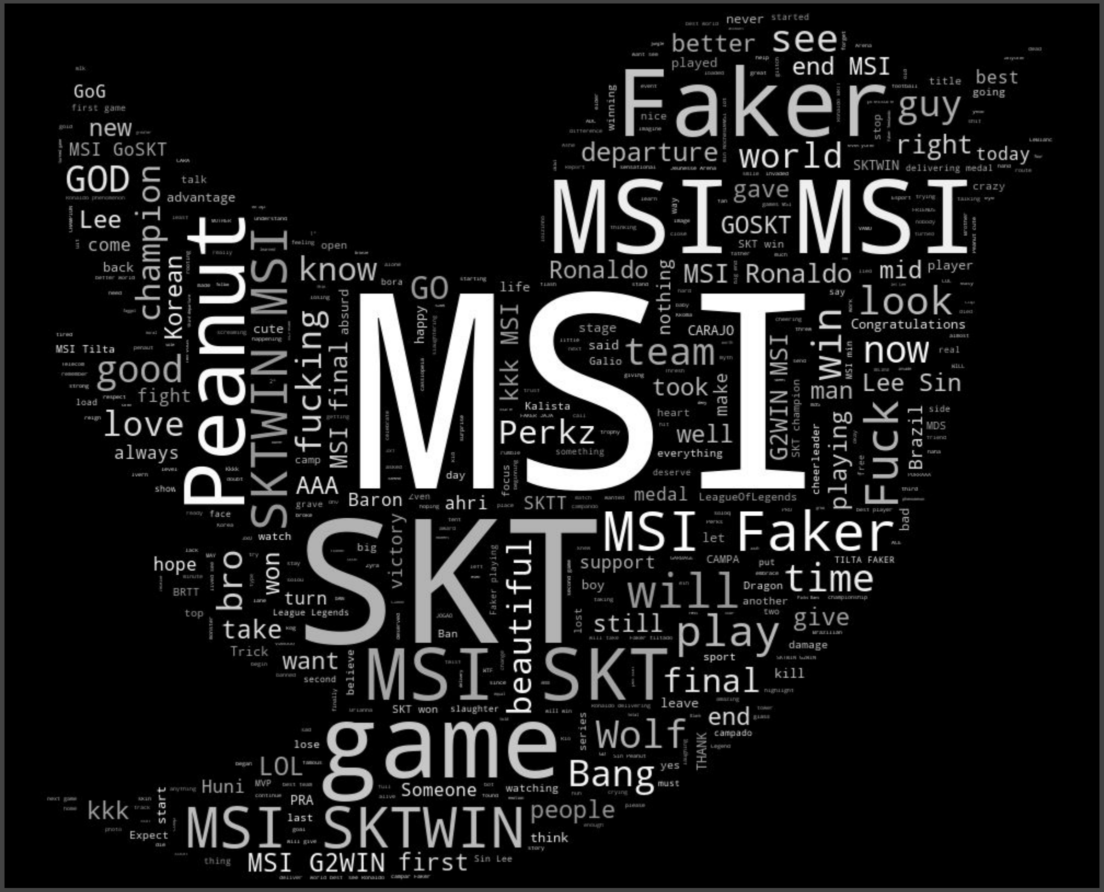


Oficina de integração 2

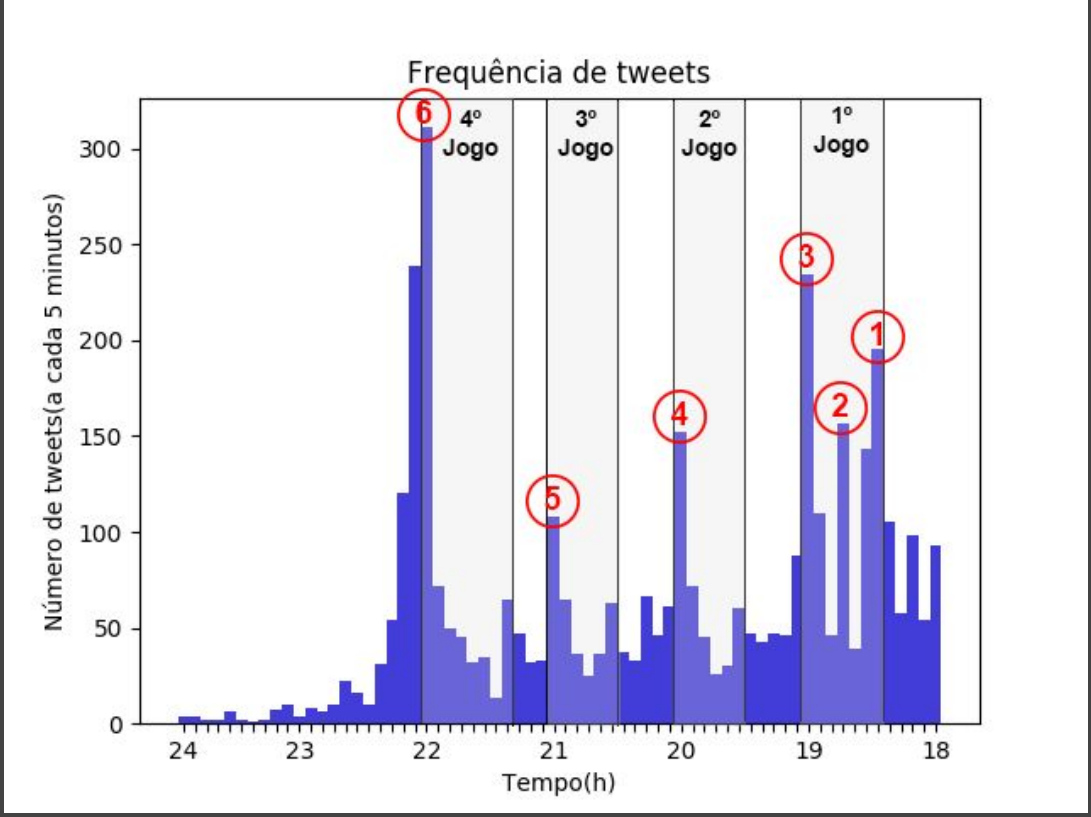


Java

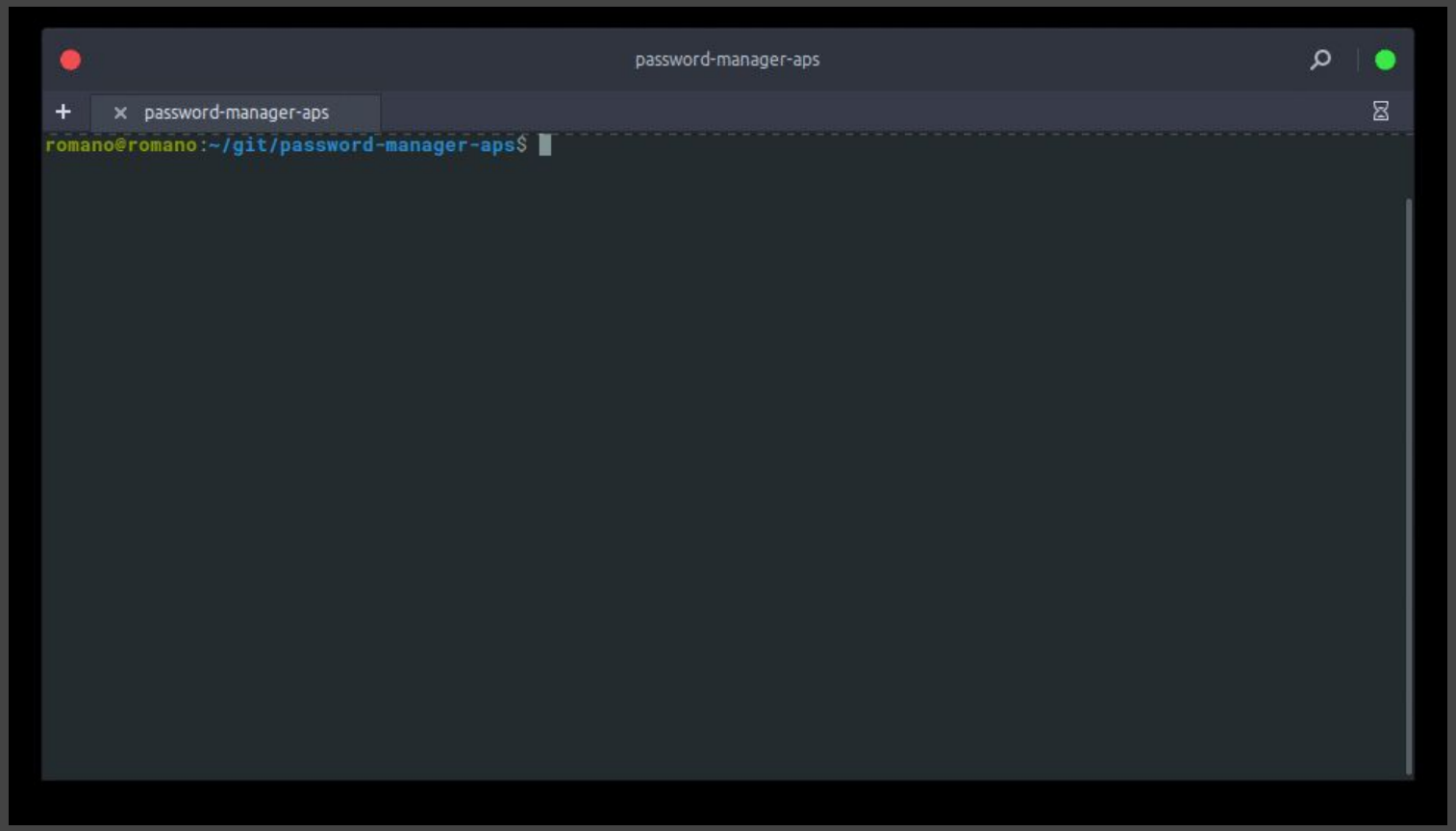
Inteligência Artificial



Inteligência Artificial



Segurança e Auditoria de sistemas



Manipulando gráficos

- **Live...**

Sua Vez !

- **Procure sobre manipulação de .csv com python.**
- **Faça um programa que crie um .csv, e outro que leia um .csv**

Obrigado!

Alguma pergunta?

Github:

https://github.com/ViniciusRomano